Authors: Z. Zhang   Y. Wei   S. Ma
         ZTE Corporation   ZTE Corporation   Google
         X. Liu   B. Rijsman
         Volta Networks   Individual

A YANG Data Model for Routing in Fat Trees (RIFT)

Abstract

   This document defines a YANG data model for the configuration and
   management of Routing in Fat Trees (RIFT) Protocol.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six
   months and may be updated, replaced, or obsoleted by other documents
   at any time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on 13 October 2022.

Table of Contents

1. Introduction

[I-D.ietf-rift-rift] introduces the protocol definition of RIFT.
This document defines a YANG data model that can be used to
configure and manage the RIFT protocol. The model is based on YANG
1.1 as defined in [RFC7950] and conforms to the Network Management
Datastore Architecture (NDMA) as described in [RFC8342]

1.1. Terminology

The terminology for describing YANG data models is found in
[RFC6020] and [RFC7950], including:

  *augment

  *container

  *choice

  *data model

  *data node

  *grouping

  *identity

*leaf

*leaf-list

*list

*module

*uses

The following abbreviations are used in this document and the
defined model:

RIFT: Routing in Fat Trees [I-D.ietf-rift-rift].

1.2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

1.3. Tree Diagrams

Tree diagrams used in this document follow the notation defined in
[RFC8340].

1.4. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model
objects are often used without a prefix, as long as it is clear from
the context in which YANG module each name is defined. Otherwise,
names are prefixed using the standard prefix associated with the
corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
rt	ietf-routing	[RFC8349]
if	ietf-interfaces	[RFC8343]
rt-types	ietf-routing-types	[RFC8294]
iana-rt-types	iana-routing-types	[RFC8294]
key-chain	ietf-key-chain	[RFC8177]

Table 1

2. Design of the Data Model

2.1. Scope of Model

The model covers RIFT [I-D.ietf-rift-rift].

This model can be used to configure and manage the RIFT protocol.
The operational state data and statistics can be retrieved by this
model. The subscription and push mechanism defined in [RFC8639] and
[RFC8641] can be implemented by the user to subscribe to
notifications on the data nodes in this model.

The model contains all the basic configuration parameters to operate
the protocol. Depending on the implementation choices, some systems
may not allow some of the advanced parameters to be configurable.
The occasionally implemented parameters are modeled as optional
features in this model. This model can be extended, and it has been
structured in a way that such extensions can be conveniently made.

The RIFT YANG module augments the /routing/control-plane-protocols/
control-plane-protocol path defined in the ietf-routing module. The
ietf-rift model defines a single instance of RIFT. Multiple
instances are instantiated as multiple control-plane protocols
instances.

2.2. Specification

This model imports and augments ietf-routing YANG model defined in
[RFC8349]. Both configuration branch and state branch of [RFC8349]
are augmented. The configuration branch covers node base and policy
configuration. The container "rift" is the top level container in
this data model. The presence of this container is expected to
enable RIFT protocol functionality.

The YANG data model defined in this document conforms to the Network
Management Datastore Architecture (NMDA) [RFC8342]. The operational
state data is combined with the associated configuration data in the
same hierarchy [RFC8407].

2.3. Overview

The RIFT YANG module defined in this document has all the common
building blocks for the RIFT protocol.

The RIFT YANG module augments the /routing/control-plane-protocols/
control-plane-protocol path defined in the ietf-routing module. The
ietf-rift model defines a single instance of RIFT. Multiple
instances are instantiated as multiple control-plane protocols
instances.

```
module: ietf-rift
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol:
    +--rw rift
       +--rw name?                         string
       +--ro level?                        level
       +--rw system-id                     system-id
       +--rw pod?                          uint32
       +--rw configured-level?             level
       +--rw overload
       |  +--rw overload?                  boolean
       |  +--rw (timeout-type)?
       |     +--:(on-startup)
       |     |  +--rw on-startup-timeout?
       |     |          rt-types:timer-value-seconds16
       |     +--:(immediate)
       |        +--rw immediate-timeout?
       |                rt-types:timer-value-seconds16
       +--ro proto-major-ver              uint8
       +--ro proto-minor-ver              uint16
       +--rw hierarchy-indications?       enumeration
       +--rw flood-reduction?             boolean
       +--rw nonce-increasing-interval?   uint16
       +--rw maximum-nonce-delta?         uint8 {nonce-delta-adjust}?
       +--rw adjusted-lifetime?
       |        rt-types:timer-value-seconds16
       +--rw rx-lie-multicast-addr
       |  +--rw ipv4?    inet:ipv4-address
       |  +--rw ipv6?    inet:ipv6-address
       +--rw tx-lie-multicast-addr
       |  +--rw ipv4?    inet:ipv4-address
       |  +--rw ipv6?    inet:ipv6-address
       +--rw lie-tx-port?                 inet:port-number
       +--rw global-link-capabilities
       |  +--rw bfd?                      boolean
       |  +--rw v4-forwarding-capable?    boolean
       +--rw rx-flood-port?               inet:port-number
       +--rw global-holdtime?
       |        rt-types:timer-value-seconds16
       +--rw tide-generation-interval?
       |        rt-types:timer-value-seconds16
       +--rw tie-security {tie-security}?
       |  +--rw (auth-key-chain)?
       |     +--:(auth-key-chain)
       |     |  +--rw key-chain?          key-chain:key-chain-ref
       |     +--:(auth-key-explicit)
       |        +--rw key-id?             uint32
       |        +--rw key?                string
       |        +--rw crypto-algorithm?   identityref
```

```
+--rw algorithm-type?             enumeration
+--rw instance-label?             uint32 {label-switching}?
+--ro hal
|  +--ro hal-value?   level
|  +--ro system-id*   system-id
+--ro miscabled-links*            linkid-type
+--rw interface* [name]
|  +--ro link-id?                   linkid-type
|  +--rw name                       if:interface-ref
|  +--rw cost?                      uint32
|  +--rw address-families*
|  |        iana-rt-types:address-family
|  +--rw advertised-source-addr
|  |  +--rw ipv4?   inet:ipv4-address
|  |  +--rw ipv6?   inet:ipv6-address
|  +--ro direction-type?            enumeration
|  +--rw security {tie-security}?
|  |  +--rw (auth-key-chain)?
|  |     +--:(auth-key-chain)
|  |     |  +--rw key-chain?        key-chain:key-chain-ref
|  |     +--:(auth-key-explicit)
|  |        +--rw key-id?           uint32
|  |        +--rw key?              string
|  |        +--rw crypto-algorithm?  identityref
|  +--rw security-checking?         enumeration
|  +--ro was-the-last-lie-accepted?  boolean
|  +--ro last-lie-reject-reason?    string
|  +--ro advertised-in-lies
|  |  +--ro you-are-flood-repeater?      boolean
|  |  +--ro not-a-ztp-offer?             boolean
|  |  +--ro you-are-sending-too-quickly?   boolean
|  +--rw link-capabilities
|  |  +--rw bfd?                    boolean
|  |  +--rw v4-forwarding-capable?  boolean
|  +--ro state                      enumeration
|  +--ro number-of-flaps?           uint32
|  +--ro last-state-change?         yang:date-and-time
+--ro neighbor* [system-id]
|  +--ro name?                       string
|  +--ro level?                      level
|  +--ro system-id                   system-id
|  +--ro pod?                        uint32
|  +--ro protocol-version?           uint16
|  +--ro sent-offer
|  |  +--ro level?           level
|  |  +--ro not-a-ztp-offer?   boolean
|  +--ro received-offer
|  |  +--ro level?                    level
|  |  +--ro not-a-ztp-offer?          boolean
```

```
|  |  +--ro best?                        boolean
|  |  +--ro removed-from-consideration?  boolean
|  |  +--ro removal-reason?              string
|  +--ro received-source-addr
|  |  +--ro ipv4?    inet:ipv4-address
|  |  +--ro ipv6?    inet:ipv6-address
|  +--ro link-id-pair* [remote-id]
|  |  +--ro local-id?          uint32
|  |  +--ro remote-id          uint32
|  |  +--ro if-index?          uint32
|  |  +--ro if-name?           if:interface-ref
|  |  +--ro address-families*  iana-rt-types:address-family
|  +--ro cost?                          uint32
|  +--ro bandwidth?                     uint32
|  +--ro received-link-capabilities
|  |  +--ro bfd?                  boolean
|  |  +--ro v4-forwarding-capable?    boolean
|  +--ro received-in-lies
|  |  +--ro you-are-flood-repeater?      boolean
|  |  +--ro not-a-ztp-offer?             boolean
|  |  +--ro you-are-sending-too-quickly?    boolean
|  +--ro tx-flood-port?                 inet:port-number
|  +--ro bfd-up?                        boolean
|  +--ro outer-security-key-id?         uint8
+--ro database
   +--ro tie* [direction-type originator tie-type tie-number]
      +--ro direction-type               enumeration
      +--ro originator                   system-id
      +--ro tie-type                     enumeration
      +--ro tie-number                   uint32
      +--ro seq?                         uint64
      +--ro origination-time?            uint32
      +--ro origination-lifetime?        uint32
      +--ro node
      |  +--ro level?                    level
      |  +--ro neighbor* [system-id]
      |  |  +--ro name?                      string
      |  |  +--ro level?                     level
      |  |  +--ro system-id                  system-id
      |  |  +--ro pod?                       uint32
      |  |  +--ro link-id-pair* [remote-id]
      |  |  |  +--ro local-id?          uint32
      |  |  |  +--ro remote-id          uint32
      |  |  |  +--ro if-index?          uint32
      |  |  |  +--ro if-name?           if:interface-ref
      |  |  |  +--ro address-families*
      |  |  |          iana-rt-types:address-family
      |  |  +--ro cost?                      uint32
      |  |  +--ro bandwidth?                 uint32
```

```
|  |   +--ro received-link-capabilities
|  |      +--ro bfd?                      boolean
|  |      +--ro v4-forwarding-capable?    boolean
|  +--ro proto-minor-ver?        uint16
|  +--ro flood-reduction?        boolean
|  +--ro hierarchy-indications
|  |  +--ro hierarchy-indications?   enumeration
|  +--ro overload-flag?           boolean
|  +--ro name?                    string
|  +--ro pod?                     uint32
|  +--ro startup-time?            yang:date-and-time
|  +--ro miscabled-links*         linkid-type
+--ro prefixes
|  +--ro prefix* [prefix]
|     +--ro prefix              inet:ip-prefix
|     +--ro metric?             uint32
|     +--ro tags*               uint64
|     +--ro monotonic-clock
|     |  +--ro prefix-sequence-type
|     |     +--ro timestamp
|     |     |     ieee802-1as-timestamp-type
|     |     +--ro transaction-id?   uint8
|     +--ro loopback?           boolean
|     +--ro directly-attached?  boolean
|     +--ro from-link?          linkid-type
+--ro positive-disagg-prefixes
|  +--ro positive-disagg-prefix*
|        [positive-disagg-prefix]
|     +--ro positive-disagg-prefix    inet:ip-prefix
|     +--ro metric?                   uint32
|     +--ro tags*                     uint64
|     +--ro monotonic-clock
|     |  +--ro prefix-sequence-type
|     |     +--ro timestamp
|     |     |     ieee802-1as-timestamp-type
|     |     +--ro transaction-id?   uint8
|     +--ro loopback?                 boolean
|     +--ro directly-attached?        boolean
|     +--ro from-link?                linkid-type
+--ro negative-disagg-prefixes
|  +--ro negative-disagg-prefix*
|        [negative-disagg-prefix]
|     +--ro negative-disagg-prefix    inet:ip-prefix
|     +--ro metric?                   uint32
|     +--ro tags*                     uint64
|     +--ro monotonic-clock
|     |  +--ro prefix-sequence-type
|     |     +--ro timestamp
|     |     |     ieee802-1as-timestamp-type
```

```
|      |        +--ro transaction-id?    uint8
|      +--ro loopback?                 boolean
|      +--ro directly-attached?        boolean
|      +--ro from-link?                linkid-type
+--ro external-prefixes
|  +--ro external-prefix* [external-prefix]
|     +--ro external-prefix     inet:ip-prefix
|     +--ro metric?             uint32
|     +--ro tags*               uint64
|     +--ro monotonic-clock
|     |  +--ro prefix-sequence-type
|     |     +--ro timestamp
|     |     |        ieee802-1as-timestamp-type
|     |     +--ro transaction-id?    uint8
|     +--ro loopback?           boolean
|     +--ro directly-attached?  boolean
|     +--ro from-link?          linkid-type
+--ro positive-ext-disagg-prefixes
|  +--ro positive-ext-disagg-prefix*
|          [positive-ext-disagg-prefix]
|     +--ro positive-ext-disagg-prefix    inet:ip-prefix
|     +--ro metric?                       uint32
|     +--ro tags*                         uint64
|     +--ro monotonic-clock
|     |  +--ro prefix-sequence-type
|     |     +--ro timestamp
|     |     |        ieee802-1as-timestamp-type
|     |     +--ro transaction-id?    uint8
|     +--ro loopback?                     boolean
|     +--ro directly-attached?            boolean
|     +--ro from-link?                    linkid-type
+--ro key-value
   +--ro key?     binary
   +--ro value?   binary

notifications:
  +---n error-set
    +--ro tie-level-error
    |  +--ro tie* [originator]
    |     +--ro direction-type?       enumeration
    |     +--ro originator            system-id
    |     +--ro tie-type?             enumeration
    |     +--ro tie-number?           uint32
    |     +--ro seq?                  uint64
    |     +--ro origination-time?     uint32
    |     +--ro origination-lifetime? uint32
    +--ro neighbor-error
      +--ro neighbor* [system-id]
         +--ro name?                     string
```

```
+--ro level?                         level
+--ro system-id                      system-id
+--ro pod?                           uint32
+--ro protocol-version?              uint16
+--ro sent-offer
| +--ro level?             level
| +--ro not-a-ztp-offer?   boolean
+--ro received-offer
| +--ro level?                         level
| +--ro not-a-ztp-offer?               boolean
| +--ro best?                          boolean
| +--ro removed-from-consideration?    boolean
| +--ro removal-reason?                string
+--ro received-source-addr
| +--ro ipv4?   inet:ipv4-address
| +--ro ipv6?   inet:ipv6-address
+--ro link-id-pair* [remote-id]
| +--ro local-id?          uint32
| +--ro remote-id          uint32
| +--ro if-index?          uint32
| +--ro if-name?           if:interface-ref
| +--ro address-families*
|         iana-rt-types:address-family
+--ro cost?                          uint32
+--ro bandwidth?                     uint32
+--ro received-link-capabilities
| +--ro bfd?                    boolean
| +--ro v4-forwarding-capable?  boolean
+--ro received-in-lies
| +--ro you-are-flood-repeater?      boolean
| +--ro not-a-ztp-offer?             boolean
| +--ro you-are-sending-too-quickly?  boolean
+--ro tx-flood-port?                 inet:port-number
+--ro bfd-up?                        boolean
+--ro outer-security-key-id?         uint8
```

2.4. RIFT configuration

The configuration data nodes cover node configuration attributes.
RIFT configurations require node base information configurations.
Some features can be used to enhance protocol, such as BFD,
flooding-reducing, community attribute.

2.5. RIFT State

The state data nodes include node, neighbor, database and kv-store
information.

2.6. Notifications

Unexpected TIE and neighbor's layer error should be notified.

3. RIFT YANG model

This module references [I-D.ietf-rift-rift], [RFC5881], [RFC6991],
[RFC8177], [RFC8294], [RFC8343], [RFC8349], [RFC8505], [IEEE8021AS].

```
<CODE BEGINS> file "ietf-rift@2022-04-11.yang"

module ietf-rift {

  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-rift";
  prefix rift;

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA Version)";
  }

  import ietf-interfaces {
    prefix "if";
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import iana-routing-types {
    prefix "iana-rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-key-chain {
    prefix "key-chain";
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }
```

```
organization
  "IETF RIFT (Routing In Fat Trees) Working Group";

contact
  "WG Web:    <http://tools.ietf.org/wg/rift/>
   WG List:   <mailto:rift@ietf.org>

   Editor:    Zheng Zhang
              <mailto:zhang.zheng@zte.com.cn>

   Editor:    Yuehua Wei
              <mailto:wei.yuehua@zte.com.cn>

   Editor:    Shaowen Ma
              <mailto:mashaowen@gmail.com>

   Editor:    Xufeng Liu
              <mailto:xufeng.liu.ietf@gmail.com>

   Editor:    Bruno Rijsman
              <mailto:brunorijsman@gmail.com>";

// RFC Ed.: replace XXXX with actual RFC number and remove
// this note

description
  "The module defines the YANG definitions for Routing in Fat
   Trees (RIFT).

   Copyright (c) 2021 IETF Trust and the persons identified as
   authors of the code.  All rights reserved.

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject
   to the license terms contained in, the Simplified BSD
   License set forth in Section 4.c of the IETF Trust's Legal
   Provisions Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC XXXX
   (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
   itself for full legal notices.

   The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
   'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
   'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
   are to be interpreted as described in BCP 14 (RFC 2119)
   (RFC 8174) when, and only when, they appear in all
   capitals, as shown here.";
```

```
revision 2022-04-11 {
  description "Initial revision.";
  reference "RFC XXXX: A YANG Data Model for RIFT.";
}

/*
 * Features
 */

feature nonce-delta-adjust {
  description
    "Support weak nonce delta adjusting which is used in security
     in section 4.4.";
  reference
    "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
}

feature label-switching {
  description
    "Support label switching for instance distinguishing in
     section 4.3.7.";
  reference
    "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
}

feature tie-security {
  description
    "Support security function described in section 4.4.3 for
     the TIE exchange.";
  reference
    "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
}

typedef system-id {
  type string {
    pattern
    '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}';
  }
  description
    "This type defines RIFT system id using pattern,
     the system id looks like: 0143.0438.0100.AeF0";
}

typedef level {
  type uint8 {
    range "0 .. 24";
  }
  default "0";
  description "The value of node level.";
```

```
    }

    typedef linkid-type {
      type uint32;
      description "This type defines the link id of an interface.";
    }

    typedef ieee802-1as-timestamp-type {
      type uint64;
      description
        "Timestamp per IEEE802.1AS. It is advertised with prefix to
         achieve mobility as described in section 4.3.3.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
         IEEE8021AS: Timing and Synchronization for Time-Sensitive
                     Applications in Bridged Local Area Networks";
    }

    /*
     * Identity
     */
    identity rift {
      base rt:routing-protocol;
      description "Identity for the RIFT routing protocol.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
    }

    /*
     * Groupings
     */

    grouping address-families {
      leaf-list address-families {
        type iana-rt-types:address-family;
        description
          "Indication which address families are up on the
           interface.";
      }
      description
        "Containing address families on the interface.";
    }

    grouping hierarchy-indications {
      leaf hierarchy-indications {
        type enumeration {
          enum "leaf-only" {
            description
              "The node will never leave the
```

```
            'bottom of the hierarchy'.";
        }
        enum "leaf-only-and-leaf-2-leaf-procedures" {
          description "This means leaf to leaf.";
        }
        enum "top-of-fabric" {
          description "The node is 'top of fabric'.";
        }
      }
      description "The hierarchy indications of this node.";
    }
    description "Flags indicating node configuration in case of ZTP";
  }

  grouping node-capability {
    leaf proto-minor-ver {
      type uint16;
      description
        "Represents the minor protocol encoding schema
         version of this node.";
    }
    leaf flood-reduction {
      type boolean;
      description
        "If this node enables the flood reduction function.";
    }
    container hierarchy-indications {
      config false;
      description "The hierarchy-indications of the node.";
      uses hierarchy-indications;
    }
    description
      "The supported capabilities of this node.";
  }

  grouping prefix-attribute {
    leaf metric {
      type uint32;
      description "The metric of this prefix.";
    }
    leaf-list tags {
      type uint64;
      description "The tags of this prefix.";
    }
    container monotonic-clock {
      container prefix-sequence-type {
        leaf timestamp {
          type ieee802-1as-timestamp-type;
          mandatory true;
```

```
      description
        "The timestamp per 802.1AS can be advertised
         with the desired prefix North TIEs.";
    }
    leaf transaction-id {
      type uint8;
      description
        "As per RFC 8505, a sequence number called a
         Transaction ID (TID) with a prefix can be
         advertised.";
      reference
        "RFC 8505: Registration Extensions for IPv6 over
         Low-Power Wireless Personal Area Network (6LoWPAN)
         Neighbor Discovery";
    }
    description
      "As described in section 4.3.3, the prefix
       sequence attribute which can be advertised
       for mobility.";
    reference
      "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
  }
  description
    "The monotonic clock for mobile addresses.";
}
leaf loopback {
  type boolean;
  description
    "Indicates if the interface is a node loopback.
     According to section 4.3.10, the node's loopback
     address can be injected into North and South
     Prefix TIEs for node reachability.";
  reference
    "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
}
leaf directly-attached {
  type boolean;
  description
    "Indicates that the prefix is directly attached,
     i.e. should be routed to even if the node is
     in overload.";
}
leaf from-link {
  type linkid-type;
  description
    "In case of locally originated prefixes,
     i.e. interface addresses this can describe which
     link the address belongs to.";
}
```

```
      description
        "The attributes of the prefix.";
    }

    grouping security {
      choice auth-key-chain {
        description
          "Key chain or explicit key parameter specification";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "key-chain name.";
          }
        }
        case auth-key-explicit {
          leaf key-id {
            type uint32;
            description "Key Identifier";
          }
          leaf key {
            type string;
            description
              "Authentication key. The length of the key may be
               dependent on the cryptographic algorithm.";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Cryptographic algorithm associated with key.";
          }
        }
      }
      description "The security parameters.";
    }

    grouping base-node-info {
      leaf name {
        type string;
        description
          "The name of this node. It won't be used as the key of node,
           just used for description.";
      }
      leaf level {
        type level;
        config false;
        description "The level of this node.";
```

```
      }
      leaf system-id {
        type system-id;
        mandatory true;
        description
          "Each node is identified via a system-id which is 64 bits
           wide.";
      }
      leaf pod {
        type uint32 {
          range "1..max";
        }
        description
          "The identifier of the Point of Delivery (PoD).
           A PoD is the self-contained vertical slice of a
           Clos or Fat Tree network containing normally only level 0
           and level 1 nodes. It communicates with nodes in other PoDs
           via the spine. Making this leaf unspecified indicates that
           the PoD is 'undefined'.";
      }
      description "The base information of a node.";
    } // base-node-info

    grouping link-capabilities {
      leaf bfd {
        type boolean;
        description
          "If this value is set to true, it means that
           BFD function is enabled on the neighbor.";
        reference
          "RFC 5881: Bidirectional Forwarding Detection (BFD)
           for IPv4 and IPv6 (Single Hop)";
      }
      leaf v4-forwarding-capable {
        type boolean;
        description
          "If this value is set to true, it means that
           the neighbor supports v4 forwarding.";
      }
      description "The features of neighbor.";
    } // link-capabilities

    grouping addresses {
      leaf ipv4 {
        type inet:ipv4-address;
        description "IPv4 address to be used.";
      }
      leaf ipv6 {
        type inet:ipv6-address;
```

```
      description "IPv6 address to be used.";
    }
    description "IPv4 and/or IPv6 address to be used.";
  }

  grouping lie-elements {
    leaf you-are-flood-repeater {
      type boolean;
      description
        "If the neighbor on this link is flooding repeater
         described in section 4.2.3.9. When this value is set
         to true, the value can be carried in exchanged
         packet.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
    }
    leaf not-a-ztp-offer {
      type boolean;
      description
        "As described in section 4.2.7. When this value is
         set to true, the flag can be carried in the LIE
         packet. When the value received in the LIE from
         neighbor, it indicates the level on the LIE MUST
         NOT be used to derive a ZTP level by the receiving
         node.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
    }
    leaf you-are-sending-too-quickly {
      type boolean;
      description
        "Can be optionally set to indicate to neighbor that
         packet losses are seen on reception based on packet
         numbers or the rate is too high. The receiver SHOULD
         temporarily slow down flooding rates. When this value
         is set to true, the flag can be carried in packet.";
    }
    description "The elements set in the LIEs.";
  } // lie-elements

  grouping link-id-pair {
    leaf local-id {
      type uint32;
      description "The local-id of link connect to this neighbor.";
    }
    leaf remote-id {
      type uint32;
      description "The remote-id to reach this neighbor.";
    }
```

```
    leaf if-index {
      type uint32;
      description "The local index of this interface.";
    }
    leaf if-name {
      type if:interface-ref;
      description "The name of this interface.";
    }
    uses address-families;
    description
      "A pair of local and remote link IDs to identify a link between
       two nodes.";
} // link-id-pair

grouping neighbor-node {
  list link-id-pair {
    key "remote-id";
    uses link-id-pair;
    description
      "The Multiple parallel links to this neighbor.";
  }
  leaf cost {
    type uint32;
    description "The cost value advertised by the neighbor.";
  }
  leaf bandwidth {
    type uint32;
    description
      "Total bits bandwith to neighbor, this will be
       normally sum of the bandwidths of all the
       parallel links.";
  }
  container received-link-capabilities {
    uses link-capabilities;
    description
      "The link capabilities advertised by the neighbor.";
  }
  description "The neighbor information indicated in node TIE.";
} // neighbor-node

grouping neighbor {
  leaf protocol-version {
    type uint16;
    description
      "Represents the protocol encoding schema version of
       this neighbor.";
  }
  container sent-offer {
    leaf level {
```

```
      type level;
      description "The level value.";
    }
    leaf not-a-ztp-offer {
      type boolean;
      description "If the neighbor needs to be offer a level.";
    }
    description
      "The level sent to the neighbor in case the neighbor
       needs to be offered.";
  }
  container received-offer {
    leaf level {
      type level;
      description "The level value.";
    }
    leaf not-a-ztp-offer {
      type boolean;
      description
        "If this interface needs to be offered a level.";
    }
    leaf best {
      type boolean;
      description
        "If level is the best level received from all
         the neighbors.";
    }
    leaf removed-from-consideration {
      type boolean;
      description
        "If the level value is considered to be used.
         If the value is not considered to be used,
         this value is set to 'TRUE'.";
    }
    leaf removal-reason {
      type string;
      description
        "The reason why this value is not considered to
         be used.";
    }
    description
      "The level offered to the interface from the neighbor.
       And if the level value is considered to be used.";
  }
  container received-source-addr {
    uses addresses;
    description
      "The source address of LIE and TIE packets from
       the neighbor.";
```

```
    } // received-offer
    uses neighbor-node;
    container received-in-lies {
      uses lie-elements;
      description "The attributes received from this neighbor.";
    }
    leaf tx-flood-port {
      type inet:port-number;
      default "915";
      description
        "The UDP port which is used by the neighbor to flood TIEs.";
    }
    leaf bfd-up {
      type boolean;
      description
        "Indication whether the link is protected by established
         BFD session.";
    }
    leaf outer-security-key-id {
      type uint8;
      description
        "As described in section 4.4.3, the received security
         key id from the neighbor.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
    }
    description "The neighbor information.";
  } // neighbor

  grouping direction-type {
    leaf direction-type {
      type enumeration {
        enum illegal {
          description "Illegal direction.";
        }
        enum south {
          description "A link to a node one level down.";
        }
        enum north {
          description "A link to a node one level up.";
        }
        enum east-west {
          description "A link to a node in the same level.";
        }
        enum max {
          description "The max value of direction.";
        }
      }
      config false;
```

```
      description "The type of a link.";
    }
    description "The type of a link.";
  } // direction-type

  grouping tie-header {
    uses direction-type;
    leaf originator {
      type system-id;
      description "The originator's system-id of this TIE.";
    }

    leaf tie-type {
      type enumeration {
        enum "node" {
          description "The node TIE.";
        }
        enum "prefix" {
          description "The prefix TIE.";
        }
        enum "positive-disaggregation-prefix" {
          description "The positive disaggregation prefix TIE.";
        }
        enum "negative-disaggregation-prefix" {
          description "The negative disaggregation prefix TIE.";
        }
        enum "pgp-prefix" {
          description "The policy guide prefix TIE.";
        }
        enum "key-value" {
          description "The key value TIE.";
        }
        enum "external-prefix" {
          description "The external prefix TIE.";
        }
        enum "positive-external-disaggregation-prefix" {
          description
            "The positive external disaggregation prefix TIE.";
        }
      }
      description "The types of TIE.";
    }

    leaf tie-number {
      type uint32;
      description "The number of this TIE";
    }

    leaf seq {
```

```
      type uint64;
      description
        "As described in section 4.2.3.1, the sequence number
         of a TIE.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
  }
  leaf origination-time {
    type uint32;
    description
      "Absolute timestamp when the TIE was generated. This can
       be used on fabrics with synchronized clock to prevent
       lifetime modification attacks.";
  }
  leaf origination-lifetime {
    type uint32;
    description
      "Original lifetime when the TIE was generated.
       This can be used on fabrics with synchronized clock to
       prevent lifetime modification attacks.";
  }

  description
    "TIE is the acronym for 'Topology Information Element'.
     TIEs are exchanged between RIFT nodes to describe parts
     of a network such as links and address prefixes.
     This is the TIE header information.";
} // tie-header

/*
 * Data nodes
 */
augment "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'rift:rift')" {
    description
      "This augment is only valid when routing protocol
       instance type is 'RIFT'.";
  }
  description "RIFT ( Routing in Fat Trees ) YANG model.";

  container rift {
    description "RIFT configuration and state data.";

    uses base-node-info;
    leaf configured-level {
      type level;
      description
        "The configured level value of this node.
```

```
          If the 'hierarchy-indications' is set to 'leaf-only'
          or 'leaf-only-and-leaf-2-leaf-procedures', this value
          means the leaf level.
          And the combination of this value and
          'hierarchy-indications' can also be used to indicate
          the maximum level value of 'top-of-fabric-level'.";
  }
  container overload {
    description
      "If the overload in TIEs can be set
       and the timeout value with according type.";
    leaf overload {
      type boolean;
      description "If the overload bit in TIEs can be set.";
    }
    choice timeout-type {
      description
        "The value of timeout timer for overloading.
         This makes sense when overload is set to 'TRUE'.";
      case on-startup {
        leaf on-startup-timeout {
          type rt-types:timer-value-seconds16;
          description
            "Node goes into overload until this timer expires
             when starting up.";
        }
      }
      case immediate {
        leaf immediate-timeout {
          type rt-types:timer-value-seconds16;
          description
            "Set overload and remove after the timeout expired.";
        }
      }
    }
  }

  leaf proto-major-ver {
    type uint8;
    config false;
    mandatory true;
    description
      "Represents protocol encoding schema major version.";
  }
  leaf proto-minor-ver {
    type uint16;
    config false;
    mandatory true;
    description
```

```
          "Represents protocol encoding schema minor version.";
      }
      uses hierarchy-indications;
      leaf flood-reduction {
        type boolean;
        description
          "If the node supports flood reduction function defined in
            section 4.2.3.8. If this value is set to 'FALSE', it
            means that the flood reduction function is disabled.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
      }
      leaf nonce-increasing-interval {
        type uint16;
        units seconds;
        description
          "The configurable nonce increasing interval.";
      }
      leaf maximum-nonce-delta {
        if-feature nonce-delta-adjust;
        type uint8 {
          range "1..5";
        }
        description
          "The configurable valid nonce delta value used for
            security. It is used as vulnerability window defined
            in section 4.4.7.
            If the nonces in received packet exceeds the range
            indicated by this value, the packet MUST be discarded.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
      }
      leaf adjusted-lifetime {
        type rt-types:timer-value-seconds16;
        units seconds;
        description
          "The adjusted lifetime may affect the TIE stability.
            Be careful to change this parameter.";
      }
      container rx-lie-multicast-addr {
        leaf ipv4 {
          type inet:ipv4-address;
          default "224.0.0.120";
          description
            "The configurable LIE receiving IPv4 multicast
              address.
              Different multicast addresses can be used for receiving
              and sending.";
        }
```

```
    leaf ipv6 {
      type inet:ipv6-address;
      description
        "The configurable LIE receiving IPv6 multicast
         address.
         Different multicast addresses can be used for receiving
         and sending.";
    }
    description
      "The configurable LIE receiving IPv4/IPv6 multicast
       address.
       Different multicast addresses can be used for receiving
       and sending.";
  }
  container tx-lie-multicast-addr {
    leaf ipv4 {
      type inet:ipv4-address;
      description
        "The configurable LIE sending IPv4 multicast address.
         Different multicast addresses can be used for receiving
         and sending.";
    }
    leaf ipv6 {
      type inet:ipv6-address;
      default "FF02::A1F7";
      description
        "The configurable LIE sending IPv6 multicast address.
         Different multicast addresses can be used for receiving
         and sending.";
    }
    description
      "The configurable LIE sending IPv4/IPv6 multicast address.
       Different multicast addresses can be used for receiving
       and sending.";
  }
  leaf lie-tx-port {
    type inet:port-number;
    default "914";
    description
      "The UDP port of LIE packet sending. The default port
       number is 914. The value can be set to other value
       associated with different RIFT instance.";
  }

  container global-link-capabilities {
    uses link-capabilities;
    description
      "The node default link capabilities. It can be overwrite
       by the configuration underneath interface and neighbor.";
```

```
        }

        leaf rx-flood-port {
          type inet:port-number;
          default "915";
          description
            "The UDP port which can be used to receive flooded
             TIEs. The default port number is 915. The value can
             be set to other value associated with different
             RIFT instance.";
        }
        leaf global-holdtime {
          type rt-types:timer-value-seconds16;
          units seconds;
          default "3";
          description "The holding time of LIE.";
        }
        leaf tide-generation-interval {
          type rt-types:timer-value-seconds16;
          units seconds;
          default "5";
          description "The TIDE generation interval.";
        }

        container tie-security {
          if-feature tie-security;
          uses security;
          description
            "As described in section 4.4.3, the security function
             used for the TIE exchange.";
          reference
            "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
        }
        leaf algorithm-type {
          type enumeration {
            enum spf {
              description "The algorithm is SPF.";
            }
            enum all-path {
              description "The algorithm is all-path.";
            }
          }
          description "The possible algorithm types.";
        }
        leaf instance-label {
          if-feature label-switching;
          type uint32;
          description
            "As per section 4.3.7, a locally significant, downstream
```

```
          assigned, interface specific label may be advertised in
          its LIEs. This value can be used to distinguish among
          multiple RIFT instances.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
    }

    container hal {
      config false;
      leaf hal-value {
        type level;
        description
          "The highest defined level value seen from all valid
           level offers received.";
      }
      leaf-list system-id{
        type system-id;
        description
          "The node's system-id of the offered level comes from.";
      }
      description
        "The highest defined level and the offered nodes set.";
    }

    leaf-list miscabled-links {
      type linkid-type;
      config false;
      description "List of miscabled links.";
    }

    list interface {
      key "name";
      leaf link-id {
        type linkid-type;
        config false;
        description "The local id of this interface.";
      }
      leaf name {
        type if:interface-ref;
        description "The interface's name.";
      }
      leaf cost {
        type uint32;
        description
          "The cost from this interface to the neighbor.";
      }
      uses address-families;
      container advertised-source-addr {
        uses addresses;
```

```
      description
        "The address used in the advertised LIE and TIE
         packets.";
    }
    uses direction-type;

    container security {
      if-feature tie-security;
      uses security;
      description
        "As described in section 4.4.3, the security function
         used for this interface.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
    }

    leaf security-checking {
      type enumeration {
        enum "no-checking" {
          description
            "The security envelop does not be checked.";
        }
        enum "permissive" {
          description
            "The security envelop checking is permissive.";
        }
        enum "loose" {
          description
            "The security envelop checking is loose.";
        }
        enum "strict" {
          description
            "The security envelop checking is strict.";
        }
      }
      description
        "The possible security checking types.
         Only one type can be set at the same time.";
    }

    leaf was-the-last-lie-accepted {
      type boolean;
      config false;
      description
        "If the most recently received LIE was accepted or
         rejected. If the LIE was rejected, the neighbor error
         notifications should be used to find the reason.";
    }
    leaf last-lie-reject-reason {
```

```
    type string;
    config false;
    description
      "Description for the reject reason of the last LIE.";
}
container advertised-in-lies {
  config false;
  uses lie-elements;
  description
    "The attributes advertised in the LIEs from
     this interface.";
}
container link-capabilities {
  uses link-capabilities;
  description
    "The interface's link capabilities.";
}
leaf state {
  type enumeration {
    enum "OneWay" {
      description "The initial state of neighbor.";
    }
    enum "TwoWay" {
      description "This means leaf to leaf.";
    }
    enum "ThreeWay" {
      description "The node is 'top of fabric'.";
    }
    enum "Multiple-Neighbors-Wait" {
      description "The node is 'top of fabric'.";
    }
  }
  config false;
  mandatory true;
  description "The hierarchy indications of this node.";
}
leaf number-of-flaps {
  type uint32;
  config false;
  description
    "The number of interface state flaps.";
}
leaf last-state-change {
  type yang:date-and-time;
  config false;
  description "Time duration in the current state.";
}

description "The interface information on this node.";
```

```
    } // list interface

    list neighbor {
      key "system-id";
      config false;
      uses base-node-info;
      uses neighbor;
      description "The neighbor's information.";
    }

    container database {
      config false;
      list tie {
        key "direction-type originator tie-type tie-number";
        description
          "A list of TIEs (Topology Information Elements).";
        uses tie-header;

        container node {
          leaf level {
            type level;
            config false;
            description "The level of this node.";
          }
          list neighbor {
            key "system-id";
            uses base-node-info;
            uses neighbor-node;
            description "The node TIE information of a neighbor.";
          }
          uses node-capability;
          leaf overload-flag {
            type boolean;
            description
              "If the overload bit in TIEs is set.";
          }
          leaf name {
            type string;
            description
              "The name of this node. It won't be used as the key
               of node, just used for description.";
          }
          leaf pod {
            type uint32;
            description
              "Point of Delivery. The self-contained vertical slice
               of a Clos or Fat Tree network containing normally
               only level 0 and level 1 nodes. It communicates with
               nodes in other PoDs via the spine. We number PoDs to
```

```
          distinguish them and use PoD #0 to denote
          'undefined' PoD.";
    }
    leaf startup-time {
      type yang:date-and-time;
      description "Startup time of the node.";
    }
    leaf-list miscabled-links {
      type linkid-type;
      config false;
        description "List of miscabled links.";
    }
    description "The node element information in this TIE.";
} // node

container prefixes {
  description
    "The prefix element information in this TIE.";
  list prefix {
    key "prefix";
    leaf prefix {
      type inet:ip-prefix;
      description "The prefix information.";
    }
    uses prefix-attribute;
    description "The prefix set information.";
  }
}
container positive-disagg-prefixes {
  list positive-disagg-prefix {
    key "positive-disagg-prefix";
    leaf positive-disagg-prefix {
      type inet:ip-prefix;
      description "The prefix information.";
    }
    uses prefix-attribute;
    description
      "The positive disaggregation prefix information.";
  }
  description "The positive disaggregation prefixes set.";
}
container negative-disagg-prefixes {
  list negative-disagg-prefix {
    key "negative-disagg-prefix";
    leaf negative-disagg-prefix {
      type inet:ip-prefix;
      description "The prefix information.";
    }
    uses prefix-attribute;
```

```
        description
          "The negative disaggregation prefix information.";
    }
    description "The negative disaggregation prefixes set.";
  }
  container external-prefixes {
    list external-prefix {
      key "external-prefix";
      leaf external-prefix {
        type inet:ip-prefix;
        description "The prefix information.";
      }
      uses prefix-attribute;
      description "The external prefix information.";
    }
    description "The external prefixes set.";
  }
  container positive-ext-disagg-prefixes {
    list positive-ext-disagg-prefix {
      key "positive-ext-disagg-prefix";
      leaf positive-ext-disagg-prefix {
        type inet:ip-prefix;
        description "The prefix information.";
      }
      uses prefix-attribute;
      description
        "The positive external disaggregation prefix
         information.";
    }
    description
      "The positive external disaggregation prefixes set.";
  }

  container key-value {
    leaf key {
      type binary;
      description "The type of key value combination.";
    }
    leaf value {
      type binary;
      description "The value of key value combination.";
    }
    description
      "The information used to distinguish a Key/Value
       pair. When the type of kv is set to 'node',
       node-element is making sense. When the type of
       kv is set to other values except 'node',
       prefix-info is making sense.";
  } // kv-store
```

```
          } // ties
          description "The TIEs information in database.";
        }// container database
      }//rift
    }//augment

    /*
     * Notifications
     */
    notification error-set {
      description "The errors notification of RIFT.";
      container tie-level-error {
        list tie {
          key "originator";
          uses tie-header;
          description "The level is undefined in the LIEs.";
        }
        description "The TIE errors set.";
      }
      container neighbor-error {
        list neighbor {
          key "system-id";
          uses base-node-info;
          uses neighbor;
          description "The information of a neighbor.";
        }
        description "The neighbor errors set.";
      }
    }
}
```

<CODE ENDS>

4. Security Considerations

The YANG module specified in this document defines a schema for data
that is designed to be accessed via network management protocols
such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF
layer is the secure transport layer, and the mandatory-to-implement
secure transport is Secure Shell (SSH) [RFC6242]. The lowest
RESTCONF layer is HTTPS, and the mandatory-to-implement secure
transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to
restrict access for particular NETCONF or RESTCONF users to a
preconfigured subset of all available NETCONF or RESTCONF protocol
operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. Writable data node represent configuration of each instance, node, interface, etc. These correspond to the following schema node:

   */rift

Modifying the configuration may cause all the RIFT neighborship to be rebuilt. For example, the configuration changing of configured-level or system-id, will lead to all the neighbor connections of this node rebuilt. The incorrect modification of authentication, except for the neighbor connection broken, will lead to the permanent connection broken. The modification of interface, will lead to the neighbor state changing. In general, unauthorized modification of most RIFT configurations will pose there own set of security risks and the "Security Considerations" in the respective reference RFCs should be consulted.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

   */rift

   */rift/interface

   */rift/neighbor

   */rift/database

The exposure of the database will expose the detailed topology of the network. Network operators may consider their topologies to be sensitive confidential data.

For RIFT authentication, configuration is supported via the specification of key-chains [RFC8177] or the direct specification of key and authentication algorithm. Hence, authentication configuration inherits the security considerations of [RFC8177]. This includes the considerations with respect to the local storage and handling of authentication keys.

5. IANA Considerations

RFC Ed.: Please replace all occurrences of 'XXXX' with the actual
RFC number (and remove this note).

This document registers a URI in the IETF XML registry [RFC3688].
Following the format in [RFC3688], the following registration is
requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-rift

Registrant Contact: The IESG

XML: N/A, the requested URI is an XML namespace.

This document also requests one new YANG module name in the YANG
Module Names registry [RFC6020] with the following suggestion:

name: ietf-rift

namespace: urn:ietf:params:xml:ns:yang:ietf-rift

prefix: rift

reference: RFC XXXX

6. Acknowledgement

The authors would like to thank Tony Przygienda, Benchong Xu
(xu.benchong@zte.com.cn), for their review, valuable comments and
suggestions.

7. References

7.1. Normative References

[I-D.ietf-rift-rift] Sharma, A., Thubert, P., Rijsman, B.,
           Afanasiev, D., and A. Przygienda, "RIFT: Routing in Fat
           Trees", Work in Progress, Internet-Draft, draft-ietf-
           rift-rift-15, 3 January 2022, <https://www.ietf.org/
           archive/id/draft-ietf-rift-rift-15.txt>.

[IEEE8021AS] "IEEE Standard for Local and Metropolitan Area Networks
           - Timing and Synchronization for Time-Sensitive
           Applications in Bridged Local Area Networks", <https://
           ieeexplore.ieee.org/document/5741898/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/
rfc2119>.

[RFC5881]   Katz, D. and D. Ward, "Bidirectional Forwarding Detection
            (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI
            10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/
            info/rfc5881>.

[RFC6020]   Bjorklund, M., Ed., "YANG - A Data Modeling Language for
            the Network Configuration Protocol (NETCONF)", RFC 6020,
            DOI 10.17487/RFC6020, October 2010, <https://www.rfc-
            editor.org/info/rfc6020>.

[RFC6241]   Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J.,
            Ed., and A. Bierman, Ed., "Network Configuration Protocol
            (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
            <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242]   Wasserman, M., "Using the NETCONF Protocol over Secure
            Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
            <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991]   Schoenwaelder, J., Ed., "Common YANG Data Types", RFC
            6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-
            editor.org/info/rfc6991>.

[RFC7950]   Bjorklund, M., Ed., "The YANG 1.1 Data Modeling
            Language", RFC 7950, DOI 10.17487/RFC7950, August 2016,
            <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040]   Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
            Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
            <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174]   Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
            2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
            May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8177]   Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J.
            Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI
            10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/
            info/rfc8177>.

[RFC8294]   Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger,
            "Common YANG Data Types for the Routing Area", RFC 8294,

DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
           BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
           <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration
           Access Control Model", STD 91, RFC 8341, DOI 10.17487/
           RFC8341, March 2018, <https://www.rfc-editor.org/info/
           rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K.,
           and R. Wilton, "Network Management Datastore Architecture
           (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018,
           <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface
           Management", RFC 8343, DOI 10.17487/RFC8343, March 2018,
           <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for
           Routing Management (NMDA Version)", RFC 8349, DOI
           10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS)
           Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446,
           August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C.
           Perkins, "Registration Extensions for IPv6 over Low-Power
           Wireless Personal Area Network (6LoWPAN) Neighbor
           Discovery", RFC 8505, DOI 10.17487/RFC8505, November
           2018, <https://www.rfc-editor.org/info/rfc8505>.

7.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
           DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of
           Documents Containing YANG Data Models", BCP 216, RFC
           8407, DOI 10.17487/RFC8407, October 2018, <https://
           www.rfc-editor.org/info/rfc8407>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard,
           E., and A. Tripathy, "Subscription to YANG

            Notifications", RFC 8639, DOI 10.17487/RFC8639, September
            2019, <https://www.rfc-editor.org/info/rfc8639>.

   [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG
              Notifications for Datastore Updates", RFC 8641, DOI
              10.17487/RFC8641, September 2019, <https://www.rfc-
              editor.org/info/rfc8641>.

Authors' Addresses

   Zheng Zhang
   ZTE Corporation

   Email: zhang.zheng@zte.com.cn

   Yuehua Wei
   ZTE Corporation

   Email: wei.yuehua@zte.com.cn

   Shaowen Ma
   Google

   Email: mashaowen@gmail.com

   Xufeng Liu
   Volta Networks

   Email: xufeng.liu.ietf@gmail.com

   Bruno Rijsman
   Individual

   Email: brunorijsman@gmail.com